UNIVERSITY OF SOUTHERN QUEENSLAND
CSC1401 – Foundation Programming (Semester 1, 2023)
Assignment 3 Specification
The HomewareCity Shopping Cart System

Due date: 19 May 2023

Weight: 20%

Type: Team-based (2 members) only one of the team members submits the assignment items on behalf of the team.

Goals and Topics

The assignment problem is straightforward. All necessary details have been supplied. The solution to the problem will use the programming concepts and strategies covered in weeks 1-10 delivered in the course. The subgoals are:

- Translating simple design into Pseudocode then Python code;
- Obtaining an advanced understanding of values, variables and lists;
- Understanding program input and output, functions and expressions;
- Understanding simple strategies like iteration, validation, sum and count;
- Understanding of advanced strategies like swapping and searching;
- Translating simple design into Python code;
- The mechanics of editing, interpreting, building and running a program;
- Testing a program;
- Commenting source code, especially Docstring on functions;
- Becoming confident and comfortable with programming in small problems.

Background

HomewareCity is a fast-growing family business in Toowoomba. In the past years, HomewareCity has served Toowoomba local community well. Aiming to promote customers' shopping experience and dealing with increasing demands raised by busy, aged, or disability customers, HomewareCity is going to introduce an online shopping facility to the community and thus, needs to develop a shopping cart system (see: http://en.wikipedia.org/wiki/Shopping cart software) to allow customers to shop online. Against a group of talented programmers, you want to win the contract to develop the system. But first, you need to develop a prototype program to demonstrate the main functionality of the system, as required by HomewareCity. Note that HomewareCity has specifically required that the program is going to be developed in Python.

The Task

In this assignment, you are required to design, implement and test a program that can be used to manage a simple shopping cart system with shopping records, which are stored in a list. Your program must provide an interactive design that allows the user to:

- create new shopping records and add them to the shopping cart system;
- display all shopping records with details;
- calculate the total cost of all the records or the records of searching outcome;

- search for specific shopping records in all records;

The program is to be implemented by Python and as a .ipynb file running on a Jupyter notebook.

Shopping cart system

Figure 1 illustrates the catalogue provided for customers before they use your program. You do not need to add the picture or table of catalogue into your program. The data in the catalogue for your program have been added to the *product_list* and *price_list* of the "Product and price lists.txt". Please directly copy the text and paste it to your .ipynb file without modification.

Catalogue

CODE	PRODUCT	PRICE	CODE	PRODUCT	PRICE	CODE	PRODUCT	PRICE	CODE	PRODUCT	PRICE
0	Salad Server Set	18.70	1	Party Serviette Holder	11.95	2	Tea Set	39.95	3	Mixing Bowl Set	49.95
4	Knife Block Set	99.95	5	Coffee Capsule Holder	29.95	6	Plastic Sensor Soap Pump	79.95	7	Storage Bucket	24.95
8	Oven Glove	9.95	9	Apron	29.95	10	Biscuit Barrel	39.95	11	Chopping Board	12.95
12	Carioca Cups	54.95	13	Soup Bowls	43.00	14	Elevate Wood Turner	19.95	15	Pasta Machine	144.95
16	Teapot	29.95	17	Cake Pop Scoop	9.95	18	Cookbook Stand	29.95	19	Chocolate Station	34.95
20	Coffee Maker	29.00	21	Pepper Mill	84.94	22	Salt Mill	84.95	23	Glass Storage Jar	4.95
24	Measuring jug	19.95	25	Kitchen Scale	39.95	26	Tenderiser	34.95	27	Pizza Docker	19.95
28	Knife Sharpener	79.95	29	Steel Cork Opener	36.95	30	Steel Garlic Press	34.95	31	Steel Can Opener	36.95
32	Stainless Steel Crank Flour Sifter	33.95	33	Mineral Stone Mortar and Pestle	74.95	34	Citrus Cather	19.95	35	Cherry & Olive Pitter	27.95
36	Multi Grater-Detachable	26.95	37	Stainless Steel Colander	44.95	38	Steel Pizza Pan	12.95	39	Pop Container	22.95

Figure 1: Catalogue

Figures 2 and 3 on the next page illustrate the shopping cart system with sample outcomes of show_record() and sorting. Please use this illustration as the reference for the following descriptions.

Shopping records

This program will hold or encapsulate data for the product code, product name, price, quantity and membership. With respect to each of the attributes, a shopping record can be represented like the following sample:

"2/Tea Set/High/39.95/1/Pick-up"

This can be deciphered as: for this shopping record, the product code is "2", the product name is Tea Set, its value is high (**When the unit price of a product is no less than $30, its value is high. Otherwise, it is low**), unit price is $39.95, quantity is 1, shipping method is "Pick-up". You should create a global variable - list - to store all such shopping records. For the sake of easy explanation, we refer to this variable by shopping_record_list in the rest of this document.

Some limitations must be imposed on the shopping records:
Product code must be "END"(case sensitive) or an integer number and only if:

- 0 <= code <=39;

Quantity must be an integer number and only if:

- 0 < quantity <=49;

Your Tasks

It is strongly suggested that the following approach is to be taken to design and implement the program.

Interactive design
You should implement and test the Interactive design of the shopping cart system for users to input value and obtain outputs. Input functions should be applied for users to enter shipping method, product code and quantity.

The add_record() Function
You should design, implement and test a function to add a shopping record to the shopping cart system. A shopping record will be added to the shopping cart system each time when all the inputs (product code, quantity and shipping method) are valid. The program will continually ask the user to input the record by three input functions until "END" (case sensitive) is input for the product code. The function handles the following tasks:

- The "product code" is obtained by the first input function. Validate if the input for "product code" is correct by using the function **is_valid_code()** described below;
- If the valid "product code" is not "END". Collect corresponding product name and price from the *product_list* and *price_list* according to "product code" for the shopping record;
- If the entered "product code" is "END". Stop the iteration and collect all the entered records and then run the **show_records() Function.**
- The "quantity" is obtained by the second input function. Validate if the input for "quantity" is correct by using the function is_valid_quantity() described below;
- The "shipping method" is obtained by the third input function. Validate inputs are "Pick-up" and "Delivery", case insensitive.
- Add the shopping record into the shopping_record_list list if all data are valid, otherwise, return an error message for each input function;

The is_valid_code() Function
You should design, implement and test a function to validate the data input for the product code attribute of a shopping record. The function should alert an error message and return false if the input is invalid, otherwise, return true.
Refer to the product code section on page 2 for what the function needs to check for validation.

The is_valid_quantity() Function
You should design, implement and test a function to validate the data input for the quantity attribute of a shopping record. The function should alert an error message and return false if the input is invalid, otherwise; return true.
Refer to the quantity section on page 2 for what the function needs to check for validation.

The show_records() Function
You should design, implement and test a function to show all existing records in the shopping cart system. The function should access the shopping_record_list and print all existing shopping records in an acceptable format as shown in Figure 2. **The length of "----" under "Product" should be 35. The maximum length of the product string is also 35**. The indentation of the output should be the same as shown in Figure 2.

```
Code   Product                              Value   Price $   Quantity   Shipping method
----   ---------------------------------    -----   -------   --------   ---------------
2      Tea Set                              High    39.95     2          Delivery
34     Citrus Cather                        Low     19.95     3          Delivery
2      Tea Set                              High    39.95     1          Pick-up
4      Knife Block Set                      High    99.95     2          Pick-up
2      Tea Set                              High    39.95     1          Delivery

Total cost is: $443.52
```

Figure 2: Shopping records and total cost

You could create some dummy shopping records manually and store them in the shopping_record_list to test this function, while other functions remain incomplete.

The total_cost() Function

You should design, implement and test a function to calculate the total cost of shopping records and display the value as a total cost as Figure 2.

You should use the following as the guideline:

- Calculate the total cost for all the records or the records of searching outcome and show the results after the list of records.
- If the customer enters "Delivery" as the shipping method, a 5% delivery fee will be added to the corresponding products. In addition, if the value of the product is high, a $2 packing fee is applied for each unit of product. No packing fee for low value products or pick-up items.
- The records for the same product may be added at a different time and the same products in one transaction may be transferred by different shipping methods (e.g., "Tea Set" in Figure 2).
- The total cost = First product unit price * quantity*(1+10%) (if shipping method is "Delivery") + quantity*2(if the delivery item's value is high) + Second product cost +……

  Take the data in figure 2 as an example: Total cost = 39.95*2*(1+10%)+2*2+19.95*3*(1+10%)+39.95*1+99.95*2+39.95*1*(1+10%)+1*2=$443.52
- control 2 digits after the decimal point for the total cost.
- If no record, the total cost will be "0".

The search_record() Function

After all the records are presented as Figure 2. You should design, implement and test a function to search the shopping records using keywords given by the user and display the search results of the shopping cart system. There is no special-order requirement for the search results.

You should use the following as the guideline:

- Let the user enter the search keyword. The program will do the search in the product names of shopping records. If the user doesn't enter any keyword and press the "Enter" key. The program will return an error message.
- The keywords for search_record() Function are case insensitive. That means if the users search "Et", the program will show all the records whose product names include "et" or "Et".

- If the product codes of two or more records are the same, the records are merged, as shown in Figure 3.
- The cost for each product is presented in the last column, control 2 digits after the decimal point). The cost includes the shipping fee and packing fee. For example, the cost for "Tea Set" in Figure 3:
  39.95*2*(1.1)+2*2+39.95*1+39.95*1*(1.1)+1*2=177.79
- Display the search results in the list as shown in Figure 3.

```
Code   Product                              Value   Price $   Quantity   Cost $
----   ---------------------------------    -----   -------   --------   ------
2      Tea Set                              High    39.95     4          395.51
4      Knife Block Set                      High    99.95     2          199.90

Total cost is: $177.79
```

Figure 3: Search results by keyword "Et"

Program Integration Test

You need to test the program for all functionality thoroughly before delivering the program to clients. The program should be running appropriately without any syntax or logic errors.

Submission

What You Need to Submit - Three Files.
For a complete submission, you need to submit three files as specified below. **Only one of the team members submits the assignment items on behalf of the team**. The assignment submission system will accept only the files with extensions specified in this section.

1. ***Meeting attendance and code contribution summary and pseudocode*** in a file saved in *.doc, .docx, .odt* format. You should first specify the registered team ID of the team (the teaching assistant will send you the ID after the team registration) and all members' names and student IDs on the top of the file.
   - **Meeting attendance and code contribution summary "**see Teamwork report and pseudocode template.docx"
   - **Pseudocode**:
     -The pseudocode should be completed before Python code.
     -The pseudocode should be succinct and accurate. Refer to week 7 course materials.
     -The pseudocode should be presented logically and clearly.
     -Ideally, good Python pseudo code enables different experienced Python programmers to complete the same program/solution independently without knowing the requirement of customers. The length of pseudocode is more than 50% of that of real code for assignment purposes. For example, python code is 50 lines, the pseudo code you write while designing the solution should be more than 25 lines. Pseudo code is not actual code, so don't just present the real python code here.

2. ***The program*** in a file saved with an *.ipynb* extension contains the source code implemented following the functional and non-functional requirements.
   The marker will only test your code in the Jupiter notebook. Please make sure your submitted Jupiter notebook file is **readable** by Anaconda installed as the "Setting up Python on your computer" section. Students cannot use any libraries or built-in functions which require an extra python package installed.
3. ***The program*** in a file saved with a *.doc/.docx/.odt* extension contains the exact same source code in the *.ipynb* file. You can just download the *.ipynb* file as *.py* file and paste code in it into the *.doc/.docx/.odt* file. If you don't submit your code in **both** *.doc/.docx/.odt* and *.ipynb* files, *or the codes* in *.doc/.docx/.odt* and *.ipynb* files are **different**. You will get a penalty (up to 10 marks).

Suggested Strategy

Plan to complete the assignment on time. Do not write all of the code in one sitting and expect that everything will be working smoothly like magic.

First step Form and register a team or check the registration status of your team if you have completed the registration process. Read assignment specification carefully. Clarify anything unclear by putting a post on the Assignment 3 Public Forum. Have your team meeting regularly (face-to-face or online zoom meeting) for discussions on assignment requirements and team working style.
Have a discussion on the options of either meeting more challenges to design the program by yourself or simply implementing the game based on the design provided. Make a decision on the options and then stick with it. Think about how to do it, how to test it, and devise high-level algorithms for each independent part of the assignment. Then, begin to type the program (with pseudocode) in incremental stages and help each other in a team.

Second step Keep working on a team basis and have your team meeting regularly. Re-read the specification, continue to refine the design of various sections to code, bring up any problems to the team for advice or to the public forum if necessary. By the end of the term, you should have had a working program and some integration tests done.

Third step Fully test the program; have another review of the source code; re-read the specification (especially marking criteria) to make sure you have done exactly what is required. Have a team meeting to discuss the experience. Write the "Statement of Completeness" and make the final submission.

Plagiarism and Academic Misconduct

USQ has zero tolerance for academic misconduct, including plagiarism and collusion. Plagiarism refers to the activity of presenting someone else's work as if you wrote it yourself. Collusion is a specific type of cheating that occurs when two or more students exceed a permitted level of collaboration on a piece of assessment. For example, identical layouts, identical mistakes, identical arguments and identical presentations in students' assignments are evidence of plagiarism and collusion. Such academic misconduct may lead to serious consequences.

Marking Criteria

You should use Table 1 as the checklist for implementation and to guide the testing of your program.

Table 1: Marking Criteria

ID	REQUIREMENTS If you don't submit your code in both .doc/.docx/.odt and .ipynb files, or the code in .doc/.docx/.odt and .ipynb files are different. You will get a penalty (up to 10 marks).	Mark
Teamwork report and pseudocode (7 marks) *You may lose corresponding parts of marks for pseudocode if you do not meet all the Functional Requirements.		
1	The "Meeting attendance" is presented clearly as required.	1
2	The "Code contribution summary" is presented clearly as required.	1
3	Pseudocode is presented in an acceptable format, and it is succinct and readable. Pseudocode is presented without redundancy.	1
4	Pseudocode for variables is well organized.	1
5	Pseudocode for functions and algorithms is presented logically and clearly.	1
6	Pseudocode for conditionals is presented logically and clearly.	1
7	Pseudocode for iterations is presented logically and clearly.	1
Functional Requirements (12 marks)		
8	The add_record() Function is designed and implemented appropriately.	1
9	The is_valid_code() Function is designed and implemented appropriately. The is_valid_quantity() Function is designed and implemented appropriately.	1
10	All the requirements about add_record() Function are met.	1
11	The show_records() Function is designed and implemented appropriately.	1
12	All the requirements about the show_records() Function are met.	1
13	The total_cost() Function is designed and implemented appropriately.	1
14	The calculation of the total cost is correct.	1
15	All the requirements about total_cost() Function are met.	1
16	The search_record() Function is designed and implemented appropriately.	1
17	The records are merged correctly and the cost for each product is calculated correctly.	1
18	All the requirements about search_record() Function are met.	1
19	All the functional requirements are met.	1

Non-functional Requirements (1 mark)		
20	The script is running free from any syntax errors. Code has been cleaned, and all testing statements (e.g., print-lining) are removed. Code in the script is indented correctly for all loops, if-else statements, and functions. Code is grouped based on common tasks. Each block of code is commented on briefly. ALL functions should be formally commented on appropriately (both style and content).	1
	Total	20